DEEP REINFORCEMENT LEARNING FOR REACTIVE IOS SPACE MANIPULATOR OPERATIONS

Matteo D'Ambrosio[1], **Lorenzo Capra**[2], **and Michèle Lavagna**[3]

[1]*Politecnico di Milano, Italy, matteo1.dambrosio@mail.polimi.it*
[2]*Politecnico di Milano, Italy, lorenzo.capra@polimi.it*
[3]*Politecnico di Milano, Italy, michelle.lavagna@polimi.it*

ABSTRACT

The application of space robotic manipulators and heightened autonomy for In-Orbit Servicing (IOS) represents a paramount pursuit for leading space agencies, given the substantial threat posed by space debris to operational satellites and forthcoming space endeavours. This work presents a guidance algorithm based on Deep Reinforcement Learning (DRL) to solve for the space manipulator path-planning during the motion synchronization phase with the mission target. The goal is the trajectory generation and control of a spacecraft equipped with a 7-Degree of Freedom (DoF) robotic manipulator, such that its end effector is stationary with respect to the target point of capture. Proximal Policy Optimization (PPO) is selected as designated DRL algorithm. The PPO algorithm generates the desired joints rates of the robotic arm, which are then integrated and passed to the controller, that is model-based feedback linearization. The agent is first trained to optimize its guidance policy generator and then tested extensively to validate the results against a simulated environment representing the motion synchronization scenario of a IOS mission.

Key words: Space manipulator; Deep Reinforcement Learning; In Orbit Servicing; Motion synchronization.

1. INTRODUCTION

The recent surging interest in advancing technologies and methodologies for In-Orbit Servicing (IOS) of satellites and space systems is motivated by the continuous expansion of space exploration and utilization, demanding for efficient and dependable approaches to repair, refuel, and reposition space assets. Spaceborne robotic systems are a key technology potentially unlocking the capability to perform these tasks, and their accurate handling and control is an essential aspect of IOS missions. The aforementioned activities are carried out either on a cooperative or an uncooperative target, and it is the latter scenario that is driving the current research field. The inherent uncertainties associated with interacting with an uncooperative target necessitate a high degree of motion control autonomy, reactivity and adaptability to the surrounding environment. The rapid progress in the field of artificial intelligence is promising substantial enhancements in the capabilities of autonomous Guidance, Navigation, and Control (GNC) within these systems. Reinforcement Learning (RL), above all, seems like a promising tool to solve complex decision making problems, formulated as Markov Decision Processes (MDP). The fusion of Neural Networks' generalization abilities with Reinforcement Learning methods has given rise to Deep Reinforcement Learning (DRL), which is extensively employed in solving planning problems for its capacity to handle high dimensional state and action spaces, as well as the possibility to cope with partially observable MDPs (POMDP).

DRL has been recently adopted to generate the trajectory to fly around a target object for its autonomous shape reconstruction in [1] and [2]. Other space applications of this methodologies were investigated by Gaudet *et al.*: adaptive guidance and control for endoatmospheric missiles is enhanced through meta-reinforcement learning in [4]; 6-Degree of Freedom (DoF) planetary landing with DRL is studied in [5]; asteroid close proximity guidance in [6].

Regarding the application of this technique to space robots, present literature is still quite scarce. In [8] a motion planning strategy for a 7-DoF space manipulator is implemented, and some of the concepts detailed in that work are also applied here. Multi-target trajectory planning is presented in [16], while control of a free-floating space robot is tackled in [18]. In the broader field of robotics, RL is being widely investigated [14], specifically for the autonomous guidance and control of robotic systems.

The aim of this work is to take a step forward the application of DRL for the trajectory planning of a space manipulator in a challenging and dynamical environment as the one of IOS missions. The results obtained demonstrate the enhanced autonomy and reactivity provided by the application of DRL in the context of the motion synchronization phase of an hypothetical IOS mission, and prove that the proposed method has the potential to be extended to a wider set of scenarios.

2. PROBLEM STATEMENT

The case study that is being analysed is now better defined. In robotic IOS missions, a key phase involves motion synchronization. The main operations performed during a robotic IOS mission are reported in a block diagram in Fig.1. During this phase, the chaser spacecraft

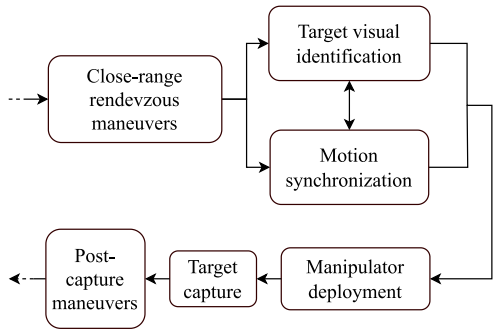

Figure 1. Mission phases.

fine-tunes its relative position and angular orientation until its end effector remains stationary relative to the target capture point. Achieving the correct relative state at the end of this closing phase is critical for the subsequent tasks of grasping and making contact. Figure 2 illustrates the problem at hand. The end effector of the space ma-

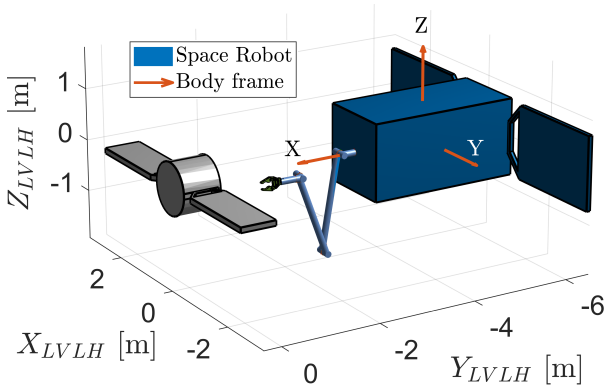

Figure 2. Motion synchronization scenario.

nipulator shall effectively track a specified grasping point on the tumbling target and follow its motion, in preparation to the subsequent activities. A generic shape for the target is selected, without loss of generality, and it is designed as a cylinder with two appendages representing solar panels. The chaser instead, is a 6-DoF spacecraft equipped with a 7-DoF redundant manipulator.
The DRL agent, that is PPO, performs the guidance and control tasks, receiving the input data from the navigation block. The work operates under the assumption of having prior knowledge of the state variables describing the scenario at every time instant, and omits the inclusion of a physical navigation block responsible for estimating these state parameters, since it is outside of the scope of this study. Consequently, the state variables are presumed to be available and are directly input into the guidance and control blocks. These blocks then generate control actions, which are subsequently applied to the system. The system, in turn, integrates the equations of motion for both the chaser and the target and provides the scenario for the next simulation step, effectively closing the feedback loop.

2.1. Space manipulator dynamics

This section provides a concise introduction to the equations of motion for a space manipulator with N degrees-of-freedom. It's important to note that, in the scope of this study, the multi-body system is described as *free-flying*, signifying that the spacecraft is actively controlled in both translation and rotation, in contrast to the *free-floating* scenario [10]. By employing the direct path approach, the spacecraft's center of mass is utilized as the representative point for translational motion, enabling the derivation of the system's kinematics and dynamics. This approach results in more streamlined equations. Using a Newton-Euler formulation, the equations of motion of the space manipulator system are computed, and they are reported in Eq. 1.

$$\boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} = \boldsymbol{\tau} \tag{1}$$

$\boldsymbol{H} \in \mathbb{R}^{(6+N)\times(6+N)}$ is the symmetric, positive -definite Generalized Inertia Matrix (GIM), $\boldsymbol{C} \in \mathbb{R}^{(6+N)\times(6+N)}$ is the Convective Inertia Matrix (CIM), containing the nonlinear contributions, the Coriolis and centrifugal forces, and $\boldsymbol{\tau} \in \mathbb{R}^{(6+N)}$ is the vector of generalized forces in the joint space. The parameter $\boldsymbol{q}$ entails the selected generalized variables, which compose the space manipulator state and are reported in Eq. 2:

$$\boldsymbol{q} = [\boldsymbol{r}_0, \boldsymbol{R}_0, \boldsymbol{q}_m]^\top = [\boldsymbol{q}_0, \boldsymbol{q}_m]^\top \tag{2}$$

where $\boldsymbol{r}_0$ is the position vector of the base spacecraft in inertial frame, $\boldsymbol{R}_0$ is the orientation of the base spacecraft with respect to the inertial frame, employing a quaternion representation, and $\boldsymbol{q}_m$ contains the joint angles of the robotic arm.
The kinematic and dynamic properties of the system are determined using the MATLAB library SPART (SPAce Robotic Toolkit) [11], a software package designed for modeling and controlling mobile-base robotic multi-body systems with efficient and recursive algorithms, taking advantage of the kinematic tree topology of the system. Additionally, for solving the equations of motion, a model of the space manipulator is constructed using the Simulink Simscape Multibody library.

2.2. Target dynamics

As the target is positioned at the origin of the LVLH reference frame, its translational motion can be disregarded, focusing instead on the rotational dynamics, which is

modeled using Euler equations in orthogonal principal axes of inertia coordinates, as in Eq. 3.

$$\boldsymbol{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\boldsymbol{I}\boldsymbol{\omega}) = \boldsymbol{M} \tag{3}$$

$\boldsymbol{I}$ is the target inertia matrix, $\boldsymbol{\omega}$ the angular velocity vector, and $\boldsymbol{M}$ is the vector of applied torques.
Once again, the equations of motion of the target are solved through a Simulink Simscape Multibody model of the target.

3. REINFORCEMENT LEARNING GUIDANCE

RL is a widely utilized tool for tackling MDPs. When combined with Neural Networks for function approximation, it becomes a potent method for addressing complex problems characterized by high-dimensionality and partial observability [15]. A cutting-edge DRL algorithm designed for problems with continuous state and action spaces, that is PPO [13], is investigated for the robotic manipulator's guidance optimization.

3.1. Proximal Policy Optimization

PPO is a state-of-the-art on-policy, model-free DRL algorithm belonging to the family of policy-gradient methods. With respect to its predecessor Trust Region Policy Optimization (TRPO) [12], PPO provides a simper implementation with higher sample efficiency, which makes for faster training without compromising reliability. As for its performance on complex, high-dimensional, and partially observable continuous control problems, PPO outperforms many of its competitors in various benchmarks and provides high training stability. PPO is based on the Actor-Critic framework [9], where the actor represents the decision-making logic of the agent (i.e. the policy π), and the critic evaluates the actions of the actor in the environment. Both actor and critic are approximated through Deep Neural Networks (DNNs) parametrized through variables θ, which are updated throughout the training process. The Actor-Critic approach is briefly described:

1. An agent is initially situated at a state s, and perceives its environment through observations o.
2. Based on o, the actor autonomously decides the action a to take, and applies it in the environment to move to a new state s'.
3. Depending on the definition of the reward $R(a_k, s_k)$, the critic evaluates the action that has been taken, and guides the parameter updates of the actor through stochastic gradient descent on a loss function.

The optimal policy in an infinite-horizon problem is found through Equation 4, and provides the agent with the maximum reward when applied in the environment.

$$\pi^* = \underset{\pi}{\text{argmax}} E_\pi \left[\sum_{k=0}^{\infty} \gamma^k R(a_k, s_k) \right] \tag{4}$$

where the discount factor $\gamma \in [0, 1]$ introduces a decay of rewards obtained distantly in time, and measures whether the agent seeks short-term or cumulative rewards.

Compared to the loss function in TRPO, PPO's clipped surrogate objective (Eqs. 5 and 6) has the advantage of limiting the policy's parameter updates by clipping the loss function, providing increased training stability.

$$p_k(\theta) = \frac{\pi_\theta(a_k|s_k)}{\pi_{\theta_{old}}(a_k|s_k)} \tag{5}$$

$$L^{CLIP}(\theta) = \\ = E_k \left[min(p_k(\theta), clip(p_k(\theta), 1-\varepsilon, 1+\varepsilon)) \right] A_k \tag{6}$$

where A_k (Equation 7) is the advantage function at timestep k, and ε is the hyperparameter defining the clipping range. The entropy loss term $S(\pi_\theta)s_k$, weighted by a hyperparameter w, is added to Eq. 6 to promote agent exploration, and encourages the actor to try a variety of different actions, without becoming too greedy towards the ones it thinks are best. Finally, the advantage function (Eq. 7) measures how advantageous taking an action a at timestep k is, with respect to simply running the current policy π_θ. The critic's job is to approximate the value function $V(s_k)$, which represents the cumulative sum of discounted rewards if only the current policy were to be run until the end of the episode.

$$A(s_k, a_k) = \left[\sum_{j=k}^{T} \gamma^{j-k} R(a_k, s_k) \right] - V(s_k) \tag{7}$$

3.2. GNC Implementation and Environment

This work presents a novel Artificial Intelligence (AI)-based autonomous guidance law for a 7-DoF redundant manipulator mounted on a Space Robot (SR), used to achieve simultaneous end-effector positioning and attitude alignment with respect to a desired state, as well as its tracking. As defined in [15], the environment in the DRL framework corresponds to everything outside of the agent's control, hence everything outside of the manipulator's guidance system is taken as part of the environment, including the remainder of the SR and the target. The SR's control system is implemented through a cou-

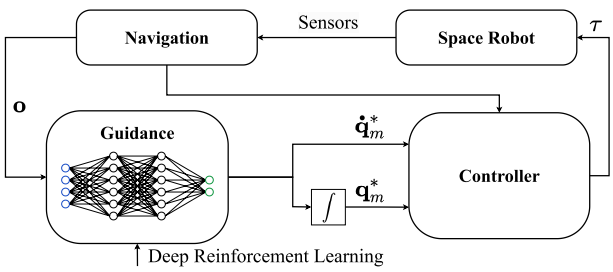

Figure 3. GNC architecture of SR

pled, nonlinear model-based feedback linearization controller, where the resulting system is controlled through

two Proportional-Derivative (PD) regulators, respectively for the base and manipulator. The base is kept at the desired synchronized state with respect to the target, while the manipulator is commanded by the DRL agent. The coupled control law is provided in Eq. 8, but being part of the DRL environment, it could be substituted with a more performant control approach with little to no retraining.

$$\boldsymbol{\tau} = \begin{Bmatrix} \boldsymbol{\tau}_0 \\ \boldsymbol{\tau}_m \end{Bmatrix} = \boldsymbol{H} \begin{Bmatrix} PD(\boldsymbol{q}_0^* - \boldsymbol{q}_0) \\ PD(\boldsymbol{q}_m^* - \boldsymbol{q}_m) \end{Bmatrix} + \boldsymbol{C}\dot{\boldsymbol{q}} \quad (8)$$

where $\boldsymbol{H}$ and $\boldsymbol{C}$ are respectively the system's 13×13 GIM and CIM [11], and $\boldsymbol{q} = [\boldsymbol{q}_0, \boldsymbol{q}_m]^\top$ collects the 6-DoFs of the base and the 7-DoFs of the manipulator. The scalar gains of the two PDs are set in Eqs. 9 and 10.

$$K_{P,0} = 0.4 \qquad K_{D,0} = 0.3 \quad (9)$$

$$K_{P,m} = 2.5 \qquad K_{D,m} = 1.25 \quad (10)$$

3.3. Action Space and Observation Space

The agent's policy, which represents the actor of the PPO implementation and provides autonomous guidance of the manipulator, receives 32 observations $\boldsymbol{o}$ (Eq. 11), and outputs 7 actions $\boldsymbol{a}$ (Eq. 12).

$$\boldsymbol{o} = [\boldsymbol{q}_m, \dot{\boldsymbol{q}}_m, \tilde{\boldsymbol{r}}, \boldsymbol{D}\tilde{\boldsymbol{C}}\boldsymbol{M}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{\omega}}]^\top \quad (11)$$

The terms in Eq. 11 correspond to the current joint angles and joint rates of the manipulator, and the errors between the current and desired end-effector state, retrieved through kinematics, and rotated in the SR's body frame. This vector is normalized before providing it to the guidance for better convergence of PPO [16]. The main benefit of using the agent only to provide manipulator guidance, is that kinematic information is sufficient in the observations, which decreases the complexity of the policy and eases convergence of the algorithm.

$$\boldsymbol{a} = \dot{\boldsymbol{q}}_m^* = [\dot{\phi}_1, \dot{\phi}_2, \dot{\phi}_3, \dot{\phi}_4, \dot{\phi}_5, \dot{\phi}_6, \dot{\phi}_7]^\top \quad (12)$$

The 7 actions produced by the agent correspond to the desired joint rates of the manipulator, and are integrated (Figure 3) such that both the desired joint angles and rates can be provided to the manipulator's PD controller [7].

3.4. Reward

Providing the agent with rewards and penalties is the sole mechanism that incentivizes the manipulator's guidance system to increase its performance. Adequate reward function design is critical as it directly impacts the convergence of the policy towards the optimal one, as well as the overall attainable performance. Since training on a sparse reward with high-dimensional state and action spaces is extremely difficult, reward shaping has been introduced through the definition of an Artificial Potential Field (APF) (Eq. 13), expanding upon [8].

$$U_k = -\tilde{r} + \frac{10}{1+\tilde{r}_{ax}} + \frac{10}{1+\tilde{r}_{tx}} + \frac{10}{1+\tilde{\theta}} \quad (13)$$

where $\tilde{r}$ is the magnitude of the error between the desired and current positions of the end-effector, $\tilde{r}_{ax}$ and $\tilde{r}_{tx}$ are the projections of $\tilde{r}$ parallel and transverse to the X-axis of the SR's body frame (Figure 2), and $\tilde{\theta}$ is the scalar error angle between the desired and current attitude of the end-effector, in axis-angle representation. The reward is given as a function of the end-effector's potential variation (ΔU) between timesteps (Eq. 14 and 15).

$$\Delta U = U_k - U_{k-1} \quad (14)$$

$$R_k = \begin{cases} \Delta U & \text{if } \Delta U \geq 0 \\ 1.5\ \Delta U & \text{if } \Delta U < 0 \end{cases} \quad (15)$$

where the $1.5\times$ multiplier discourages the end-effector from moving along equipotential surfaces. A bonus sparse reward of $+0.01$ is provided while $\tilde{r}_{ax}$, $\tilde{r}_{tx}$, and $\tilde{\theta}$ are simultaneously below a desired threshold.

4. TRAINING AND RESULTS

Before proceeding with training, the initial conditions and the DRL hyperparameters are introduced. The scenario is that of a target spacecraft tumbling around its major inertia axis. The SR's state is kept synchronized with that of the target, such that they spin together and any relative motion between the desired end-effector state and the SR is minimized. The SR is positioned along the angular momentum ($\boldsymbol{L}_T$) of the target at a nominal distance of 5 m, and its angular velocity is set as in Eq. 16 for synchronization purposes.

$$\boldsymbol{\omega}_0 = [\boldsymbol{\omega}_T \cdot \hat{\boldsymbol{L}}_T, 0, 0]^\top \quad (16)$$

The nominal initial manipulator state is found in Eq. 17.

$$\boldsymbol{q}_m = [0, 285, 0, 210, 0, 75, 0]^\top \text{ deg} \quad \dot{\boldsymbol{q}}_m = 0 \quad (17)$$

To increase the robustness of the agent, and to show that it can adapt to conditions that haven't strictly been trained on, the nominal initial conditions of the target object and of the SR are randomized at the start of each simulation:

- Target's major-axis spin rate $\boldsymbol{\omega}_T \in [-3, 3]$ deg/s.
- Initial manipulator joint angles are separately perturbed by a random value $\delta\phi_i \in [-15, 15]$ deg.
- Desired end-effector state is randomized on the whole SR-facing side of the target, both in terms of position and attitude.
- Magnitude of distance between SR and target is perturbed by a random value $\delta d \in [-25, 25]$ cm.

Simulations are only terminated if the manipulator's configuration becomes singular, to prevent the DRL algorithm from breaking down due to mathematical issues. With regards to the PPO hyperparameters, the sample time of the agent is set to 0.3 s as a trade-off between computational expense, convergence, and reactivity of

Table 1. Actor & Critic Networks.

Layers	Actor neurons	Critic neurons
Input	32	32
1^{st} hidden	300	300
2^{nd} hidden	300	300
3^{rd} hidden	300	300
Output	14	1
Learning Rate	1e-5	1e-5
Activation	$tanh$	$tanh$

the SR. The actor and critic are represented through two Feedforward Neural Networks (FNNs), with hyperparameters in Table 1. A stochastic policy is used to increase agent exploration [18], hence the actor's 14 outputs (Table 1) represent respectively the mean and standard deviation of each desired joint velocity. The remainder of the PPO hyperparameters are selected among typical values: the clipping factor $\varepsilon = 0.2$, the discount factor $\gamma = 0.99$, the entropy loss weight $w = 0.01$, the minibatch size is 128, and the training epochs are 4. The agent is trained for 7500 episodes, each of 420 s duration, for a total of 10.5 M timesteps (Figure 4). Apart from the

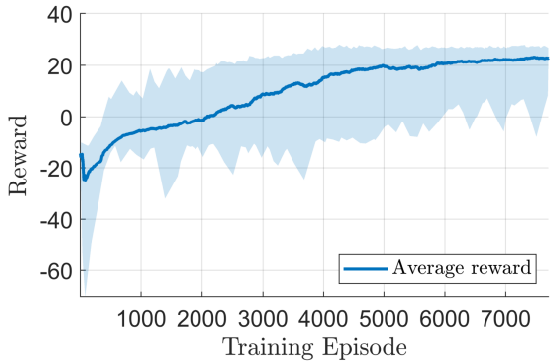

Figure 4. Average training episode reward.

initial episodes where the agent uses network parameters that have only just been initialized, the average reward grows over time until it converges around a final value.

4.1. Agent performance

The agent's success in a simulation is defined as its ability to keep the end-effector within a selected tolerance from the desired state, in terms of both position and orientation, consecutively for at least $t_{min} = 30$ s. This differs with respect to what is currently done in the majority of literature, where, once the end-effector enters the selected threshold for the first time, the episode is considered successful and the simulation is terminated. In such a highly dynamic scenario, the latter approach does not prove that the end-effector's state can remain synchronized with that of the grasping position, and would artificially increase the agent's performance in the environment.

The minimum error thresholds that guarantee a 100% success rate of the agent, and that represent its performance baseline, are $\tilde{r}_{ax}, \tilde{r}_{tx} < 5$ cm and $\tilde{\theta} < 5$ deg. These results are confirmed through the Montecarlo analysis in Figures 5 and 6, which show that regardless of the grasping point's location in the target's body frame, the agent can successfully synchronize the manipulator's end-effector with the desired state, for consecutive periods that are much higher than t_{min}.

Through a deeper analysis of Figure 6, the average time

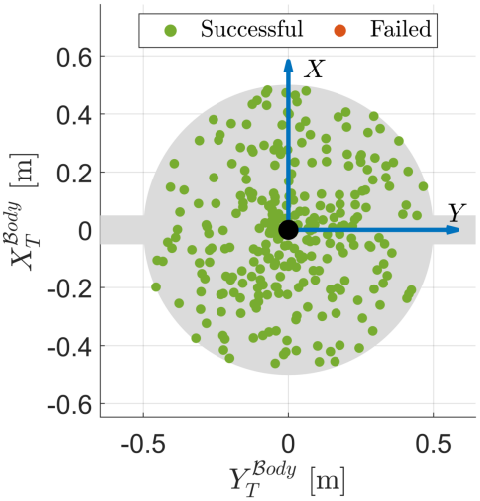

Figure 5. Grasping location and success correlation.

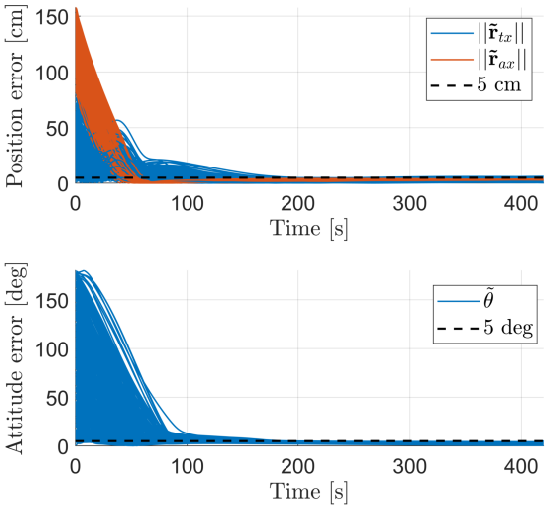

Figure 6. Monte Carlo analysis over 500 episodes.

that the end-effector takes to successfully converge to the desired state is found to be 103 s, and in any case, no episodes take longer than 219 s to accomplish the objec-

tive, which is approximately half of the complete episode duration. These values are driven by the randomized initial configuration between manipulator and grasping point, and increase proportionally to the range of motion that needs to be carried out by the robotic arm. Additionally, the average consecutive time that the end-effector stays within the selected error tolerances is 312 s, corresponding to 74% of the total episode duration. This confirms that once the end-effector converges to the desired state, it does not manifest a largely oscillatory behavior.
Building on the few studies found in the literature, this work demonstrates that the proposed AI based robotic arm guidance strategy, when applied to a 7-DoF redundant manipulator which has a randomized positioning and attitude alignment goal for extended periods of time, reliably provides performance in the order of centimeters and degrees. These results show an improvement of what is currently found in literature: in [17] the guidance of a 7-DoF manipulator is trained to achieve an end-effector positioning goal, whereas its attitude is neglected; in [8], a 7-DoF manipulator is trained to accomplish both a positioning and attitude alignment objective, but only the first 6 of 7 joints are controlled, since the end-effector is symmetrical around the last joint's rotation axis.

4.2. Agent robustness

The need for highly reactive , adaptive, and autonomous systems anticipated for future close-proximity operations, has been one of the driving factors towards the introduction of AI based methods into spacecraft GNC. Despite being new, the recent applications of DRL in the space field have emerged as promising strategies towards the generation of highly adaptive agents, that can handle unforeseen conditions that have not strictly been trained on, with significant increases towards mission robustness. These capabilities have been shown to be intrinsic to the use of deep neural networks, and if achieved, would provide many benefits supporting the addition of AI into classic GNC systems. To give some preliminary insight into why using such approaches could be advantageous, the agent's limits and generalization capabilities are stressed in two scenarios that it has not been trained to handle.
To this extent, errors in the spin rate synchronization around the target's rotation axis are added, to see whether the agent can adapt to this new scenario without further training. The difference with the previous case, in which the grasping point is static with respect to the SR, is that the end-effector now needs to track a moving point and synchronize its motion with it, maintaining a constant attitude. The maximum spin rate error between the base of the SR and the target is taken from the COMRADE study [3], where the requirement for the angular rate control error is set to $||\boldsymbol{\omega}_0 - \boldsymbol{\omega}_T|| < 0.5$ deg/s. Hence, the SR's angular velocity is perturbed each episode by a random value $\omega_{err} \in [-0.5, 0.5]$ deg/s, as in Eq. 18. An overview of this new scenario is provided in Figure 7.

$$\boldsymbol{\omega}_0 = [\boldsymbol{\omega}_T \cdot \hat{\boldsymbol{L}}_T + \omega_{err}, 0, 0]^\top \tag{18}$$

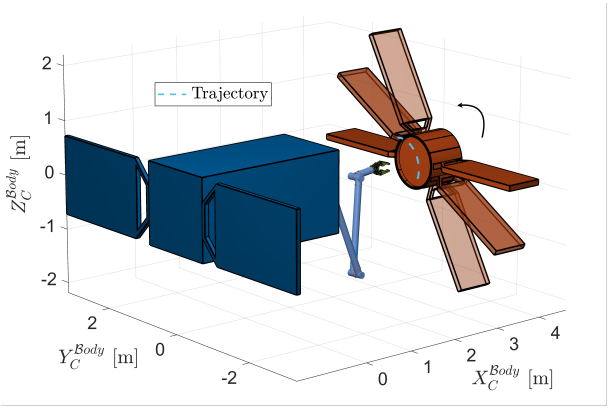

Figure 7. Trajectory of desired end-effector position.

A Monte Carlo analysis is conducted to evaluate the agent's performance over 500 testing episodes. In these conditions that the agent has never experienced during training, the success rate, defined in the same way as in the previous section, drops to 94%. These results show that despite a small decrease in performance, the agent is robust to errors in the attitude synchronization, and is capable of tracking a moving position in time. Referring to Figure 8, it can be seen that the episode failures do not show a correlation to ω_{err}, since many episodes are successful even when the synchronization error between the SR and target is high in magnitude. Instead, the failures of the agent are more so tied to the initial configuration between the manipulator and grasping point, and are located primarily in the third quadrant.

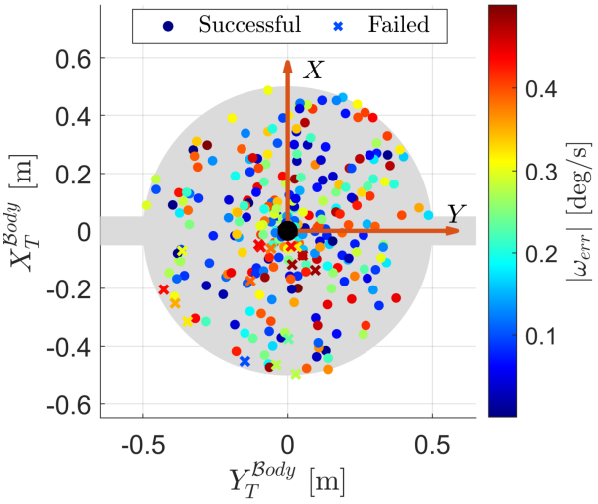

Figure 8. Synchronization error correlation to success.

For a more thorough comparison of the agent's behavior with and without errors in the SR's base attitude synchronization, the distribution of two performance indicators is reported in Figures 9 and 10: the first figure

shows how the time of the end-effector's first successful entry in the thresholds is distributed among episodes, whereas the second figure shows the distribution of the maximum consecutive time that the end-effector remains inside of the threshold, in each episode. Overall, even when the agent is subjected to a new environment that has not been trained on, its behavior is quite similar to the one it demonstrates in nominal conditions. The main difference is found in terms of outliers in the distributions, which recur more often when synchronization errors are present. Despite this similarity in results, better and more robust performance could be obtained by directly training the agent to handle the more complex environment, where possible.

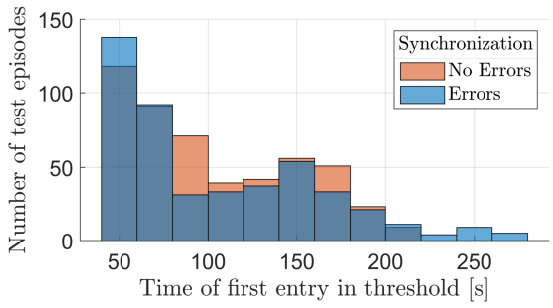

Figure 9. First end-effector entry in threshold.

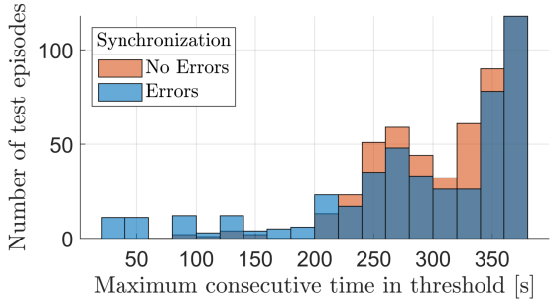

Figure 10. End-effector consecutive time in threshold.

To better understand how and why the episodes are failing, a sensitivity analysis on the definition of episode success is carried out. The end-effector's error thresholds $\tilde{r}_{ax}, \tilde{r}_{tx} < 5$ cm and $\tilde{\theta} < 5$ deg, that need to be guaranteed consecutively for at least $t_{min} = 30$ s, have been selected arbitrarily and in real scenarios would be heavily mission-dependent. Figure 11 shows the variation of the success rate over 500 episodes, when these values are changed, in two distinct cases.

1. The curves associated to the left axis show how the success rate varies in function of the thresholds on $\tilde{r}_{ax}$, $\tilde{r}_{tx}$, and t_{min}, while keeping the one on $\tilde{\theta}$ fixed.
2. The curves associated to the right axis show how the success rate varies in function of the thresholds on $\tilde{\theta}$ and t_{min}, while keeping the ones on $\tilde{r}_{ax}$ and $\tilde{r}_{tx}$ fixed.

From Figure 11, it can be seen that in both analyses, varying t_{min} has negligible effects on the success rate, which is explained by the fact that in the majority of cases, the agent can keep the end-effector's errors low for consecutive periods much longer than t_{min}. By increasing the threshold on $\tilde{\theta}$ from its baseline value of 5 deg, the success rate remains unchanged, signifying that the end-effector's attitude is not the main factor limiting performance. Differently, by increasing the end-effector's positioning thresholds, the success rate starts to increase, making this value act as the main bottleneck in the obtained performance. These results are determined by a

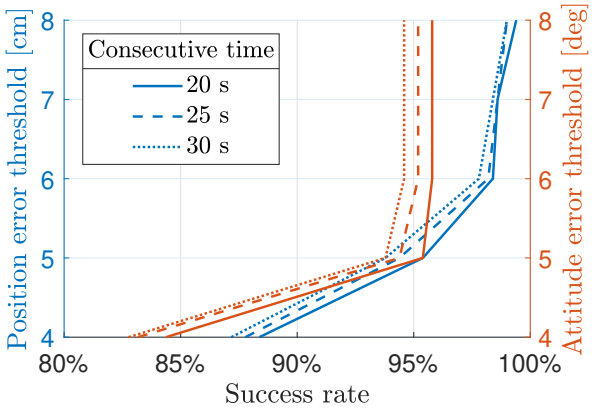

Figure 11. Success sensitivity to thresholds.

combination of different effects: firstly, the simple PD controller that is used to control the system after feedback linearization cannot guarantee null steady-state errors, which is a first factor impacting the convergence of the end-effector towards its final desired state; secondly, the agent's sample time of 0.3 s, coupled with the integration of the actor's outputs, may also be reducing the agent's maximum performance, especially once the end-effector errors have been reduced below the baseline threshold values.

A final test is conducted to further stress the agent's generalization capabilities, when applied to a Target that is larger than the one used during training. Specifically, the agent has been trained to correctly position and align the end-effector in front of a Target of 50 cm radius, and is instead asked to complete the same randomized objective, but on a Target of 150 cm radius. The agent's performance is evaluated over 500 testing episodes, and its success rate is shown in Figure 12. The results show that as the goal position of the end-effector moves outside of the area where it has been trained, the performance drops significantly. Despite this, it has been found that no episodes fail below a radius of 64.5 cm, which shows that the agent can adapt to a condition that it has not experienced during training, and achieve the objective on a Target that is at most 28% larger than the one used in training. To confirm these results in a statistical sense, 200 additional testing episodes are conducted, randomizing the goal position within a radius of 64.5 cm from the center of the Target, and the success rate of the agent remains at 100%.

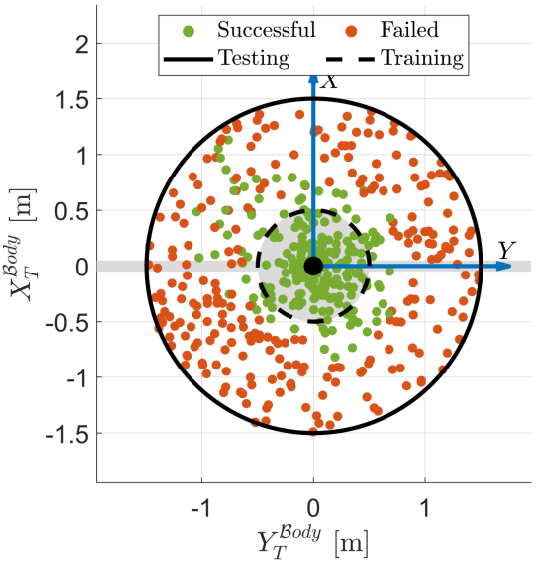

Figure 12. Generalization to larger target.

5. CONCLUSIONS

This work proposes a novel autonomous guidance algorithm for the manipulator of a free-flying space robot, allowing to automatically synchronize the end-effector with a desired state fixed to the uncooperative target spacecraft, in a hypothetical IOS mission. The problem is formulated as a Partially Observable Markov Decision Process (POMDP), and solved through the state-of-the-art PPO algorithm. A FNN provides the guidance of the manipulator in real-time based on values retrieved through the navigation system, which is not implemented, and its outputs are provided to a model-based feedback linearization controller, which couples the control laws of the base of the servicer and its manipulator. After the training process, the agent successfully reaches a randomized end-effector state objective, in a highly randomized environment, with a 100% success rate, keeping its errors in terms of position and attitude below thresholds of 5 cm and 5 deg for lengthy consecutive periods. Without any further training, the same agent is found to be robust to errors in the attitude synchronization between SR and target, and can also complete the same objective on a target that is at most 28% larger than the one used during training. Future extensions of similar approaches could obtain better results by using more performant control systems, that guarantee null steady-state errors, or by decreasing the sample time of the agent. The latter would have to be tuned based on the frequency of the values provided by the navigation filters.

Despite the literature on this topic being new and with many shortcomings, the results produced in this work convey that DRL should be investigated further, as a prospective solution to a wider set of robotic IOS scenarios.

REFERENCES

[1] Brandonisio, A., Capra, L., & Lavagna, M. 2023, Advances in Space Research

[2] Capra, L., Brandonisio, A., & Lavagna, M. 2023, Advances in Space Research, 71

[3] Colmenarejo, P., Branco, J., Santos, N., et al. 2018, 69th International Astronautical Congress (IAC), 1

[4] Gaudet, B. & Furfaro, R. 2023, arXiv:2109.03880

[5] Gaudet, B., Linares, R., & Furfaro, R. 2020, Advances in Space Research, 65, 1723

[6] Gaudet, B., Linares, R., & Furfaro, R. 2020, Acta Astronautica, 171, 1

[7] Kumar, V., Hoeller, D., Sundaralingam, B., Tremblay, J., & Birchfield, S. 2020, arXiv:2011.06332

[8] Li, Y., Li, D., Zhu, W., et al. 2022, Aerospace, 9

[9] Mnih, V., Badia, A. P., Mirza, M., et al. 2016, arXiv:1602.01783

[10] Papadopoulos, E., Aghili, F., Ma, O., & Lampariello, R. 2021, Frontiers in Robotics and AI, 8

[11] Romano, M., Virgili-Llop, J., & Ii, J. V. D. 2016, 6th International Conference on Astrodynamics Tools and Techniques

[12] Schulman, J., Levine, S., Moritz, P., Jordan, M. I., & Abbeel, P. 2015, arXiv:1502.05477, 1889

[13] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. 2017, arXiv:1707.06347

[14] Singh, B., Kumar, R., & Singh, V. P. 2022, Artificial Intelligence Review, 55, 945

[15] Sutton, R. S. & Barto, A. G. 2018, Reinforcement learning: An Introduction (Westchester Publishing Services), 526

[16] Wang, S., Zheng, X., Cao, Y., & Zhang, T. 2021, IEEE International Conference on Intelligent Robots and Systems, 3724

[17] Wu, Y. H., Yu, Z. C., Li, C. Y., et al. 2020, Aerospace Science and Technology, 98

[18] Yan, C., Zhang, Q., Liu, Z., Wang, X., & Liang, B. 2018, IEEE International Conference on Robotics and Biomimetics